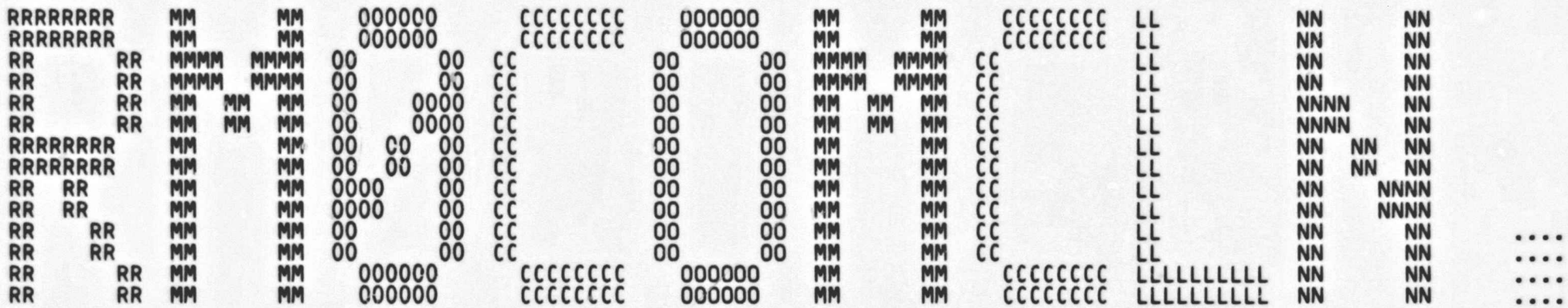
```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM   SSS
RRR         RRR  MMM   MMM   MMM   SSS
RRR         RRR  MMM   MMM   MMM   SSS
RRRRRRRRRRR      MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**RMOCOMCLN



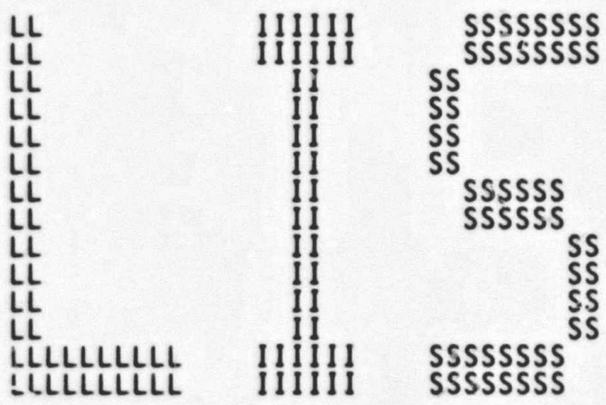


Table of contents

```
0000     1          $BEGIN  RMOCOMCLN,000,RM$RMS0,<COMMON CLEAN UP CONN-DISCONN>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;++
0000    28 ; Facility: rms32
0000    29 ;
0000    30 ; Abstract:
0000    31 ;                   this module provides four entry points to provide common
0000    32 ;                   clean up on connect - disconnect.
0000    33 ;
0000    34 ; Environment:
0000    35 ;                   star processor running starlet exec.
0000    36 ;
0000    37 ; Author: L F Laverdure,          Creation Date: 31-MAR-1977
0000    38 ;
0000    39 ; Modified By:
0000    40 ;
0000    41 ;       V03-008 JWT0160          Jim Teague                29-Feb-1984
0000    42 ;                   Complete the anticipated removal of RM$DEALLEFN.
0000    43 ;
0000    44 ;       V03-007 SHZ0001          Stephen H. Zalewski       3-Feb-1983
0000    45 ;                   Make routine RM$DEALLEFN a NOP in anticipation of
0000    46 ;                   removing it altogether.
0000    47 ;
0000    48 ;       V03-006 KPL0008          Peter Lieberwirth         29-Apr-1983
0000    49 ;                   Always deallocate IRAB journaling structures.
0000    50 ;
0000    51 ;       V03-005 KBT0367          Keith B. Thompson         11-Oct-1982
0000    52 ;                   Deallocate irab and asb seperatly
0000    53 ;
0000    54 ;       V03-004 JWH0002          Jeffrey W. Horn           20-Sep-1982
0000    55 ;                   Rename RLB$W_OWNER to RLB$L_OWNER.
0000    56 ;
0000    57 ;       V03-003 KBT0322          Keith B. Thompson         9-Sep-1982
```

```
0000     58 ;                  Remove all SO sharing code
0000     59 ;
0000     60 ;       V03-002 KBT0202         Keith B. Thompson       23-Aug-1982
0000     61 ;                  Reorganize psects and fix rev. history of jwh0001
0000     62 ;
0000     63 ;       V03-001 JWH0001         Jeffrey W. Horn         19-May-1982
0000     64 ;                  Add call to RM$DSCJNL to get rid of journaling BDB and
0000     65 ;                  buffer.
0000     66 ;
0000     67 ;--
0000     68 ;
0000     69
```

```
0000     71          .SBTTL  DECLARATIONS
0000     72
0000     73 ;
0000     74 ; Include Files:
0000     75 ;
0000     76
0000     77 ;
0000     78 ; Macros:
0000     79 ;
0000     80
0000     81          $BLBDEF
0000     82          $GBPBDEF                        ; global buffer pointer block
0000     83          $IFBDEF
0000     84          $DEVDEF
0000     85          $IRBDEF
0000     86          $CSHDEF
0000     87          $RLSDEF
0000     88          $RLBDEF
0000     89          $FABDEF
0000     90          $RABDEF
0000     91          $PIODEF
0000     92          $IMPDEF
0000     93          $PSLDEF
0000     94          $RMSDEF
0000     95
0000     96 ;
0000     97 ; Equated Symbols:
0000     98 ;
0000     99
0000    100 ;
0000    101 ; Own Storage:
0000    102 ;
0000    103
```

```
0000    105          .SBTTL  RM$DISCOMMON - COMMON CLEANUP ON CONNECT-DISCONNECTT ROUTINES
0000    106
0000    107 ;++
0000    108 ; RM$DISCOMMON
0000    109 ;
0000    110 ;
0000    111 ;       RM$DISCOMMONSUC - sets r0 to rm$_suc and falls thru to rm$discommon
0000    112 ;       RM$DISCOMMON    - checks for network disconnect and falls thru to
0000    113 ;                         rm$comclnup
0000    114 ;       RM$COMCLNUP     - return all bdb's and buffers for stream and fall thru
0000    115 ;                         to rm$ccln1 r0 already pushed onto stack
0000    116 ;       RM$CCLN1        - deallocate irab and zero isi and irab table entry
0000    117 ;
0000    118 ; Calling sequence:
0000    119 ;
0000    120 ;       bsbw    rm$discommonsuc
0000    121 ;       bsbw    rm$discommon
0000    122 ;       brw     rm$comclnup
0000    123 ;       bsbw    rm$ccln1
0000    124 ;
0000    125 ; Input Parameters:
0000    126 ;
0000    127 ;       r11     impure area address
0000    128 ;       r10     ifab address
0000    129 ;       r9      irab address
0000    130 ;       r8      rab address
0000    131 ;       r0      status code (except for entry at rm$discommonsuc)
0000    132 ;
0000    133 ; Implicit Inputs:
0000    134 ;
0000    135 ;   for entry at rm$discommon:  irb$v_pap_conn
0000    136 ;   for entry at rm$comclnup:   irb$b_bcnt & bdb chain
0000    137 ;   for entry at rm$ccln1:      irb$l_irab_lnk
0000    138 ;
0000    139 ; Output Parameters:
0000    140 ;
0000    141 ;       r0      status code
0000    142 ;       r7      set from irb$b_mode
0000    143 ;       r9      zeroed
0000    144 ;       ap      set from irb$l_arglst
0000    145 ;       r1-r6   destroyed
0000    146 ;
0000    147 ; Implicit Outputs:
0000    148 ;
0000    149 ;       rab$w_isi zeroed
0000    150 ;       rab$l_stv possibly updated
0000    151 ;       irab, its bcb's, bdb's and related buffers deallocated
0000    152 ;
0000    153 ; Completion Codes:
0000    154 ;
0000    155 ;       standard rms. if an error occurs it will replace
0000    156 ;       the status code input in r0, otherwise the code input
0000    157 ;       in r0 will be used.
0000    158 ;
0000    159 ; Side Effects:
0000    160 ;
0000    161 ;       none
```

```
0000   162 ;--
0000   163
```

```
                        0000   165
                        0000   166 ;++
                        0000   167 ;
                        0000   168 ; entry point to set r0 = rm$_suc and fall thru into rm$discommon
                        0000   169 ;
                        0000   170 ;--
                        0000   171
                        0000   172 RM$DISCOMMONSUC::
                        0000   173         RMSSUC
                        0003   174
                        0003   175 ;++
                        0003   176 ;
                        0003   177 ; entry point to check for network disconnect and fll thru into rm$comclnup
                        0003   178 ;
                        0003   179 ;--
                        0003   180
                        0003   181 RM$DISCOMMON::
              50   DD   0003   182         PUSHL   R0                      ; save status
                        0005   183
                        0005   184 ;
                        0005   185 ;       check for process-permanent file cleanup
                        0005   186 ;
                        0005   187
           69   22   E1   0005   188         BBC     #IRB$V_PPF_IMAGE,(R9),-
                03        0008   189                 1$                      ; branch if not indirect acc to ppf
              00DD   31   0009   190         BRW     IND_FILE                ; branch
           6A   2E   E1   000C   191 1$:      BBC     #IFB$V_PPF_INPUT,(R10),-
                10        000F   192                 10$                     ; branch if not sys$input
           2401 8F   B0   0010   193         MOVW    #1+<^A/$/@8>,-
00000000'9F             0014   194                 @#PIO$GT_ENDSTR         ; reset eod string to '$'
                02   8A   0019   195         BICB2   #1@PIO$V_EOD,-
00000000'9F             001B   196                 @#PIO$GW_STATUS         ; and clear eod flag
                        0020   197 10$:
           69   3C   E5   0020   198         BBCC    #IRB$V_DAP_CONN,(R9),-
                0D        0023   199                 BIOCHK                  ; not net then continue
                        0024   200
                        0024   201 ;++
                        0024   202 ;
                        0024   203 ;  perform network disconnect function
                        0024   204 ;
                        0024   205 ;--
                        0024   206
                        0024   207 NTDISC:
           6B   04   E0   0024   208         BBS     #IMP$V_IORUNDOWN,(R11),-
                09        0027   209                 BIOCHK                  ; branch if i/o rundown in progress
              FFD5'   30   0028   210         BSBW    NT$DISCONNECT           ; disconnect at remote node
           03   50   E8   002B   211         BLBS    R0,BIOCHK               ; branch if successful
           6E   50   D0   002E   212         MOVL    R0,(SP)                 ; save error code
                        0031   213
                        0031   214 BIOCHK:
                05   E0   0031   215         BBS     #IFB$V_BIO,-
        0B 22   AA        0033   216                 IFB$B_FAC(R10),NOBUFF   ; branch if block i/o
           23   AA   95   0036   217         TSTB    IFB$B_ORGCASE(R10)      ; is this seq f.o. ?
                0C   13   0039   218         BEQL    RM$COMCLNUP             ; if yes, no lock bdb to return
           6A   33   E0   003B   219         BBS     #IFB$V_NORECLK,(R10),-
                08        003E   220                 RM$COMCLNUP             ; branch if no locking
                2A   11   003F   221         BRB     RTNBLB                  ; return the lock blb.
```

```
                     0041   222 NOBUFF:
                     0041   223
                     0041   224 ;
                     0041   225 ; This is the return the block i/o bdb.  This will branch into cache
                     0041   226 ; and release to get rid of it.  Because the bdb was not counted as
                     0041   227 ; a buffer when allocated, the avlcl count is bumped so cache will
                     0041   228 ; just find it and take it.  Cache will also set the bdb$w_users count
                     0041   229 ; to 1 so that release is not upset for the relative and isam orgs.
                     0041   230 ; The only reason cache and release are used at all is because if
                     0041   231 ; someone is out there doing asynch multistreaming with block i/o,
                     0041   232 ; this should prevent us from returning a bdb in use, because cache
                     0041   233 ; will look for one with a users count of 0, and the block i/o code
                     0041   234 ; sets the users count to 1 when using it.
                     0041   235 ;
                     0041   236
0084 CA    B6        0041   237         INCW    IFB$W_AVLCL(R10)        ; want to fake out cache so it
                     0045   238                                         ; doesn't try to free one.
     05    11        0045   239         BRB     BIO                     ; branch to return it.
                     0047   240
                     0047   241 ;++
                     0047   242 ;
                     0047   243 ; entry point to return all blb's, bdb's and buffers for this stream.
                     0047   244 ; status already pushed onto stack.  this is error path from rm$bdballoc
                     0047   245 ; to return bdb's, blb's allocated before failure.  irab will also be
                     0047   246 ; deallocated before returning to user so no other structures can be present.
                     0047   247 ; this is strictly error path on connect operation.  lock blb will not have
                     0047   248 ; been allocated.
                     0047   249 ;
                     0047   250 ;--
                     0047   251
                     0047   252 RM$COMCLNUP::
                     0047   253
                     0047   254 ;++
                     0047   255 ;
                     0047   256 ;       return bdb's used by this stream
                     0047   257 ;
                     0047   258 ;--
                     0047   259
  54 A9    97        0047   260         DECB    IRB$B_BCNT(R9)          ; decrement buffer count
     48    19        004A   261         BLSS    CHKGBL                  ; branch if no more
                     004C   262 BIO:     $CACHE  VBN=#0,SIZE=#0,-
                     004C   263                 FLAGS=<NOREAD>          ; get any BDB.
  EE 50    E9        0056   264         BLBC    R0,RM$COMCLNUP          ; failed so go around again
53   01    D0        0059   265 RTNBDB: MOVL    #RLS$M_RETURN,R3        ; set return flag
   FFA1'    30        005C   266         BSBW    RM$RELEASE              ; release bdb & buffer
  03 50     E8        005F   267         BLBS    R0,RTNJNL               ; go check for blb release
6E   50     D0        0062   268         MOVL    R0,(SP)                 ; save error code
                     0065   269
00000000'EF   16      0065   270 RTNJNL: JSB     RM$DSCJNL               ; clean up IRAB journal structures
                     006B   271
                     006B   272 ;++
                     006B   273 ;
                     006B   274 ; check for locking and return blb's if so.
                     006B   275 ;
                     006B   276 ;--
                     006B   277
                     006B   278 RTNBLB:
```

```
                D8 6A    33    E0  006B   279          BBS     #IFB$V_NORECLK, (R10), RM$COMCLNUP ; branch back if no locking.
                         02    10  006F   280          BSBB    RTNBLBS                    ; Return a BLB.
                         D4    11  0071   281          BRB     RM$COMCLNUP                ; Loop to get any more.
                               0073   282 RTNBLBS:
                               0073   283          ASSUME  BLB$L_FLNK EQ 0
                               0073   284          ASSUME  BLB$L_BLNK EQ 4
              54   0098 CA     DE  0073   285          MOVAL   IFB$L_BLBFLNK(R10), R4     ; get list head.
                   50   54     D0  0078   286          MOVL    R4,R0                      ; save for end test.
                54   04 A4     D0  007B   287 10$:     MOVL    4(R4), R4                  ; get blb element.
                   54   50     D1  007F   288          CMPL    R0, R4                     ; back at list head?
                        09     13  0082   289          BEQL    20$                        ; it's a bug if we are.
                     24 A4     D5  0084   290          TSTL    BLB$L_LOCK_ID(R4)          ; is this one in use?
                        F2     12  0087   291          BNEQ    10$                        ; NEQ it is, get another.
                      FF74'    31  0089   292          BRW     RM$RETBLB                  ; and return it.
                               05  008C   293          RSB                                ; Return.
                               008D   294 20$:
                               008D   295          RMSPBUG FTL$_NOBLB
                               0094   296
                               0094   297 CHKGBL:
                27 69    36    E1  0094   298          BBC     #IRB$V_GBLBUFF,(R9),RTNRLB ; Branch if no gbpb, blb allocated.
                         7E    D4  0098   299          CLRL    -(SP)                      ; Init pass counter.
                54   40 AA     DE  009A   300 10$:     MOVAL   IFB$L_BDB_FLNK(R10),R4     ; Get list head address.
                   50   54     D0  009E   301          MOVL    R4, R0                     ; Save for end test.
                               00A1   302
                               00A1   303          ASSUME  IFB$L_BDB_BLNK  EQ      <IFB$L_BDB_FLNK + 4>
                               00A1   304          ASSUME  GBPB$C_BLINK    EQ      4
                               00A1   305
                54   04 A4     D0  00A1   306 20$:     MOVL    4(R4), R4                  ; Scan backwards.
                   50   54     D1  00A5   307          CMPL    R4, R0                     ; Back at head?
                        12     13  00A8   308          BEQL    30$                        ; Continue if so.
                               00AA   309          ASSUME  <GBPB$C_BID & 1> EQ 1
                  F3 08 A4     E9  00AA   310          BLBC    GBPB$B_BID(R4), 20$        ; Keep looking if not GBPB.
                     0C A4     B5  00AE   311          TSTW    GBPB$W_USERS(R4)           ; Is use count zero?
                        EE     12  00B1   312          BNEQ    20$                        ; Keep looking if not.
                      FF4A'    30  00B3   313          BSBW    RM$RETGBPB                 ; Return the GBPB.
                        BB     10  00B6   314          BSBB    RTNBLBS                    ; Return the BLB.
                DE 6E    00    E3  00B8   315          BBCS    #0, (SP), 10$              ; Branch if 1st pass.
                   5E    04    C0  00BC   316 30$:     ADDL2   #4, SP                     ; Remove pass counter.
                               00BF   317
                               00BF   318 ;++
                               00BF   319 ;
                               00BF   320 ;  unlock all locked records for this stream and deallocate all unused rlb's
                               00BF   321 ;
                               00BF   322 ;--
                               00BF   323
                      FF3E'    30  00BF   324 RTNRLB: BSBW    RM$UNLOCKALL
              53   59    38    C1  00C2   325          ADDL3   #IRB$L_RLB_LNK,R9,R3       ; get rlb list head addr in r3
                   5C    53    D0  00C6   326          MOVL    R3,AP                      ; copy it
                               00C9   327          ASSUME  RLB$L_LNK EQ 0
                54   6C    D0  00C9   328 40$:     MOVL    (AP),R4                    ; get next rlb addr
                        19     13  00CC   329          BEQL    60$                        ; branch if no more
                     10 A4     D5  00CE   330          TSTL    RLB$L_OWNER(R4)            ; in use?
                        0F     12  00D1   331          BNEQ    55$                        ; branch if yes
                6C   64    D0  00D3   332          MOVL    RLB$L_LNK(R4),(AP)         ; remove rlb from chain
                52   1C    D0  00D6   333          MOVL    #RLB$C_BLN,R2              ; set rlb length
                        53     DD  00D9   334          PUSHL   R3                         ; save space header addr
                      FF22'    30  00DB   335          BSBW    RM$RETSPC1                 ; deallocate rlb
```

```
                08   BA   00DE   336          POPR    #^M<R3>                 ; restore space header addr
                E7   11   00E0   337          BRB     40$                     ; go get next rlb
          5C    54   D0   00E2   338 55$:     MOVL    R4,AP                   ; rlb in use - copy addr
                E2   11   00E5   339          BRB     40$                     ; go get next rlb
                0F   11   00E7   340 60$:     BRB     RTNIRB
```

```
                     00E9    342
                     00E9    343 ;++
                     00E9    344 ;
                     00E9    345 ;  this is a disconnect of an indirectly connected irab (i.e., for the image)
                     00E9    346 ;
                     00E9    347 ;  just zero isi and clear eof if unit record device
                     00E9    348 ;
                     00E9    349 ;--
                     00E9    350
                     00E9    351 IND_FILE:
02 A8    B4          00E9    352         CLRW    RAB$W_ISI(R8)           ; clear isi
   00    E1          00EC    353         BBC     #DEV$V_REC,-
41 6A                00EE    354                 IFB$L_PRIM_DEV(R10),EXIT; branch if not unit record device
                     00F0    355         CSB     #IRB$V_EOF,(R9)         ; reset eof flag
   3B    11          00F4    356         BRB     EXIT
```

```
                     00F6   358
                     00F6   359 ;++
                     00F6   360 ;
                     00F6   361 ; entry point for when no bdb's or buffers allocated.
                     00F6   362 ; simply deallocate the irab and zeroes isi.
                     00F6   363 ;
                     00F6   364 ;--
                     00F6   365
                     00F6   366 RM$CCLN1::
          50   DD    00F6   367          PUSHL   R0                          ; save error code
                     00F8   368                                             ; find this irab in irab chain
                     00F8   369
      53   5A   D0   00F8   370 RTNIRB: MOVL    R10,R3                      ; get ifab addr
                     00FB   371          ASSUME  IRB$L_IRAB_LNK EQ IFB$L_IRAB_LNK
  56  1C A3   D0   00FB   372 10$:     MOVL    IRB$L_IRAB_LNK(R3),R6       ; get next irab
      59   56   D1   00FF   373          CMPL    R6,R9                       ; is this the one?
           05   13   0102   374          BEQL    20$                         ; branch if yes
      53   56   D0   0104   375          MOVL    R6,R3                       ; move ptr to other reg
           F2   11   0107   376          BRB     10$                         ; & keep searching
                     0109   377
                     0109   378 ;
                     0109   379 ; got the irab  -  close up chain and deallocate the irab
                     0109   380 ;
                     0109   381
1C A3   1C A6   D0   0109   382 20$:     MOVL    IRB$L_IRAB_LNK(R6),IRB$L_IRAB_LNK(R3)
                     010E   383
                     010E   384 ;
                     010E   385 ;  restore the user's mode and arg list pointer from the irab before
                     010E   386 ;  deallocating it.
                     010E   387 ;
                     010E   388
57   0A A9   9A   010E   389          MOVZBL  IRB$B_MODE(R9),R7
5C   18 A9   D0   0112   390          MOVL    IRB$L_ARGLST(R9),AP
                     0116   391
                     0116   392 ;
                     0116   393 ; deallocate asb and irab
                     0116   394 ;
                     0116   395
      53   5A   D0   0116   396          MOVL    R10,R3                      ; get space header page
54    14 A9   D0   0119   397          MOVL    IRB$L_ASBADDR(R9),R4        ; get asb addr
           06   13   011D   398          BEQL    30$                         ; just in case we don't have one?
         FEDE'  30   011F   399          BSBW    RM$RETBLK                   ; deallocate irab
      53   5A   D0   0122   400          MOVL    R10,R3                      ; restore header page
      54   59   D0   0125   401 30$:     MOVL    R9,R4                       ; get irab addr
         FED5'  30   0128   402          BSBW    RM$RETBLK                   ; deallocate irab
51    1C AB   D0   012B   403          MOVL    IMP$L_IRABTBL(R11),R1       ; get irab table addr
           07   10   012F   404          BSBB    ZAPCOM                      ; zero isi & isi table entry
           01   BA   0131   405 EXIT:   POPR    #^M<R0>                     ; restore status
                05   0133   406          RSB                                ; & return
```

```
                     0134   408
                     0134   409 ;++
                     0134   410 ;
                     0134   411 ; subroutine to clear the ifab or irab table
                     0134   412 ; entry for the ifab or irab whose address is in r9.
                     0134   413 ; also zeros the ifi or isi and r9
                     0134   414 ;
                     0134   415 ; inputs:
                     0134   416 ;       r11     impure area address
                     0134   417 ;       r9      ifab/irab address
                     0134   418 ;       r8      fab/rab address
                     0134   419 ;
                     0134   420 ; outputs:
                     0134   421 ;       ifab/irab table pointer zeroed.
                     0134   422 ;       ifab/irab address in r9 zeroed.
                     0134   423 ;       fab$w_ifi/rab$w_isi zeroed
                     0134   424 ;       r0-r2 destroyed
                     0134   425 ;
                     0134   426 ;--
                     0134   427
                     0134   428 RM$ZAPIFI::
51   18 AB   DE      0134   429         MOVAL   IMP$L_IFABTBL(R11),R1   ; get ifab table addr
                     0138   430 ZAPCOM:
52   04 A1   DE      0138   431         MOVAL   4(R1),R2                ; leave r1 pointing to link
                     013C   432                                         ; point r2 to 1st entry
                     013C   433
50   20 AB   3C      013C   434         MOVZWL  IMP$W_ENTPERSEG(R11),R0 ; # entries per table segment.
   59   82   D1      0140   435 10$:    CMPL    (R2)+,R9                ; is this desired entry?
        08   13      0143   436         BEQL    20$                     ; branch if yes
     F8 50   F5      0145   437         SOBGTR  R0,10$                  ; keep trying
   51   61   D0      0148   438         MOVL    (R1),R1                 ; next segment
        EB   11      014B   439         BRB     ZAPCOM
                     014D   440
                     014D   441 ;
                     014D   442 ; this is the sought-for table entry  -  zero it
                     014D   443 ;
                     014D   444
        72   D4      014D   445 20$:    CLRL    -(R2)
                     014F   446         ASSUME  RAB$W_ISI EQ FAB$W_IFI
02 A8   84           014F   447         CLRW    RAB$W_ISI(R8)           ; zero isi (or ifi)
22 AB   B7           0152   448         DECW    IMP$W_NUM_IFABS(R11)    ; decrement count of allocated ifabs
   59   D4           0155   449         CLRL    R9                      ; zero ifab or irab address
        05           0157   450         RSB
                     0158   451
                     0158   452         .END
```

```
$$.PSECT_EP                     = 00000000
$$.TMP                          = 00000004
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
BIO                               0000004C R      01
BIOCHK                            00000031 R      01
BLB$L_BLNK                      = 00000004
BLB$L_FLNK                      = 00000000
BLB$L_LOCK_ID                   = 00000024
CHKGBL                            00000094 R      01
CSH$M_NOBUFFER                  = 00000008
CSH$M_NOREAD                    = 00000004
DEV$V_REC                       = 00000000
EXIT                              00000131 R      01
FAB$W_IFI                       = 00000002
FTL$_NOBLB                      = FFFFFFDF
GBPB$B_BID                      = 00000008
GBPB$C_BID                      = 00000015
GBPB$L_BLINK                    = 00000004
GBPB$W_USERS                    = 0000000C
IFB$B_FAC                       = 00000022
IFB$B_ORGCASE                   = 00000023
IFB$L_BDB_BLNK                  = 00000044
IFB$L_BDB_FLNK                  = 00000040
IFB$L_BLBFLNK                   = 00000098
IFB$L_IRAB_LNK                  = 0000001C
IFB$L_PRIM_DEV                  = 00000000
IFB$V_BIO                       = 00000005
IFB$V_NORECLK                   = 00000033
IFB$V_PPF_INPUT                 = 0000002E
IFB$W_AVLCL                     = 00000084
IMP$L_IFABTBL                   = 00000018
IMP$L_IRABTBL                   = 0000001C
IMP$V_IORUNDOWN                 = 00000004
IMP$W_ENTPERSEG                 = 00000020
IMP$W_NUM_IFABS                 = 00000022
IND_FILE                          000000E9 R      01
IRB$B_BCNT                      = 00000054
IRB$B_MODE                      = 0000000A
IRB$L_ARGLST                    = 00000018
IRB$L_ASBADDR                   = 00000014
IRB$L_IRAB_LNK                  = 0000001C
IRB$L_RLB_LNK                   = 00000038
IRB$V_DAP_CONN                  = 0000003C
IRB$V_EOF                       = 00000021
IRB$V_GBLBUFF                   = 00000036
IRB$V_PPF_IMAGE                 = 00000022
NOBUFF                            00000041 R      01
NT$DISCONNECT                     ********   X    01
NTDISC                            00000024 R      01
PIO$GT_ENDSTR                     ********   X    01
PIO$GW_STATUS                     ********   X    01
PIO$V_EOD                       = 00000001
RAB$W_ISI                       = 00000002
RLB$C_BLN                       = 0000001C
RLB$L_LNK                       = 00000000
RLB$L_OWNER                     = 00000010
RLS$M_RETURN                    = 00000001
RM$BUG                            ********   X    01
RM$CACHE                          ********   X    01
RM$CCLN1                          000000F6 RG     01
RM$COMCLNUP                       00000047 RG     01
RM$DISCOMMON                      00000003 RG     01
RM$DISCOMMONSUC                   00000000 RG     01
RM$DSCJNL                         ********   X    01
RM$RELEASE                        ********   X    01
RM$RETBLB                         ********   X    01
RM$RETBLK                         ********   X    01
RM$RETGBPB                        ********   X    01
RM$RETSPC1                        ********   X    01
RM$UNLOCKALL                      ********   X    01
RM$ZAPIFI                         00000134 RG     01
RTNBDB                            00000059 R      01
RTNBLB                            0000006B R      01
RTNBLBS                           00000073 R      01
RTNIRB                            000000F8 R      01
RTNJNL                            00000065 R      01
RTNRLB                            000000BF R      01
ZAPCOM                            00000138 R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMSO	00000158 (344.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:00.89
Command processing	114	00:00:00.71	00:00:05.02
Pass 1	347	00:00:11.72	00:00:35.48
Symbol table sort	0	00:00:01.53	00:00:02.32
Pass 2	93	00:00:02.35	00:00:05.38
Symbol table output	11	00:00:00.13	00:00:01.72
Psect synopsis output	2	00:00:00.03	00:00:00.16
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	598	00:00:16.56	00:00:50.98

The working set limit was 1650 pages.
65097 bytes (128 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1211 non-local and 16 local symbols.
452 source lines were read in Pass 1, producing 14 object records in Pass 2.
31 pages of virtual memory were used to define 30 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	19
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	26

1366 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RMOCOMCLN/OBJ=OBJ$:RMOCOMCLN MSRC$:RMOCOMCLN/UPDATE=(ENH$:RMOCOMCLN)+EXECML$/LIB+LIB$:RMS/LIB
```

RM0EXTRMS
LIS

RM0DIRSCN
LIS

RM0CRECOM
LIS

RM0FABCHK
LIS

RM0CHKSUM
LIS

RM0FWASET
LIS

RM0EXTEND
LIS

RM0IFISI
LIS

RM0JOURNL
LIS

RM0FSETI
LIS

RM0FSET
LIS

RM0COMCLN
LIS

RM0DUMMY
LIS

RM0FILFNC
LIS